7. What Are Some Common SSRF Attack Scenarios?

Common SSRF (Server-Side Request Forgery) attack scenarios involve exploiting a vulnerable server to make unauthorized requests to internal systems or external services. These attacks can be targeted at various aspects of an application's architecture, including accessing internal resources, exploiting cloud metadata, or bypassing firewalls. Below are some of the most **common SSRF attack scenarios**:

---

1. Accessing Internal Services (Port Scanning)

- **Scenario:** An attacker uses SSRF to send requests to **internal services** that are usually not accessible from the outside.
  - **How it works:** The attacker exploits a web application that allows users to submit URLs or IPs for server-side requests. The attacker crafts a request that points to an internal service (e.g., `http://localhost:3306`, `http://internal-db.local`).
  - **Impact:** This can reveal internal services (like databases, admin interfaces, or APIs) that are supposed to be protected by firewalls. The attacker might gain unauthorized access to internal systems or use this information for further exploitation (e.g., database access or privilege escalation).
- **Example Attack:**
  An attacker may use SSRF to discover and exploit a web server running an internal database, allowing them to exfiltrate sensitive information.

---

2. Exploiting Cloud Metadata Services

- **Scenario:** The attacker targets cloud metadata services (e.g., AWS, Google Cloud) using SSRF to gain access to cloud-specific information, such as credentials, metadata, or IAM roles.
  - **How it works:** Many cloud services expose metadata at a specific IP (e.g., `http://169.254.169.254` for AWS). If an application allows requests to arbitrary URLs, an attacker can send SSRF requests to this IP to retrieve metadata about the cloud instance, including **API keys**, **IAM credentials**, and **instance data**.
  - **Impact:** If the attacker is able to retrieve **cloud credentials**, they can escalate their privileges and access critical resources in the cloud environment.
- **Example Attack:**
  An attacker sends an SSRF request to `http://169.254.169.254/latest/meta-data/` from a vulnerable web application. This exposes instance metadata, including sensitive cloud credentials like access keys, which the attacker can then use to interact with the cloud services and access storage, databases, or other resources.

---

3. **Bypassing IP Whitelisting and Firewalls**

- **Scenario:** SSRF can be used to bypass network-level access restrictions such as **IP whitelisting** or **firewalls**.
  - **How it works:** If an internal service is protected by IP-based restrictions (e.g., only allowing requests from certain IP addresses), an attacker can exploit SSRF to **send requests** on behalf of the server to services that would otherwise be inaccessible from the external network.
  - **Impact:** Attackers can use the vulnerable application to interact with internal systems, bypassing network security controls and gaining access to resources that should be isolated.
- **Example Attack:**
  An attacker may bypass a firewall that restricts direct access to an internal service by exploiting SSRF. For instance, the attacker targets an internal API that is only accessible by IP whitelisting, and uses SSRF to send requests to that API through the vulnerable server.

---

4. **Triggering Internal System Vulnerabilities**

- **Scenario:** SSRF can be used to trigger vulnerabilities in internal services or systems that the attacker would otherwise be unable to access.
  - **How it works:** The attacker uses SSRF to reach an internal service and attempts to exploit a known vulnerability, such as a **remote code execution (RCE)** or **SQL injection** flaw.
  - **Impact:** This can lead to internal system compromise, privilege escalation, and even complete control over vulnerable systems.
- **Example Attack:**
  The attacker sends an SSRF request to an internal web service that has an RCE vulnerability. By injecting malicious payloads through SSRF, the attacker gains control over the internal system and escalates their privileges.

---

5. **Accessing Sensitive Internal Files or Configuration Data**

- **Scenario:** Attackers can use SSRF to access sensitive **internal files** or **configuration data** on the server.
  - **How it works:** The attacker makes SSRF requests to internal resources (e.g., `/etc/passwd`, `/etc/shadow`, configuration files) that store sensitive information.
  - **Impact:** By accessing internal files, attackers can obtain sensitive data, like **user credentials**, **API keys**, or **service configurations**, that could help them further compromise the system.
- **Example Attack:**
  The attacker may send an SSRF request that targets `http://localhost/etc/passwd`, hoping to retrieve the system's password file, which could help with escalating their attack to gain root access.

---

6. **Redirecting Requests to External Malicious Servers**

- **Scenario:** An attacker may use SSRF to redirect requests from the vulnerable server to an external server controlled by the attacker.
  - **How it works:** The attacker provides a URL pointing to a malicious external server. This could be a **phishing** server, a **malicious endpoint**, or a **Command and Control (C2) server**.
  - **Impact:** By tricking the server into making requests to malicious resources, attackers can **exfiltrate data**, install **malware**, or **phish** other users.
- **Example Attack:**
  An attacker exploits SSRF to send a request to a malicious external server that logs the server's IP and response headers, helping the attacker gain insights into the internal server's configuration or network.

---

7. DoS (Denial of Service) via Resource Exhaustion

- **Scenario:** SSRF can be used to perform a **Denial of Service (DoS)** attack by exhausting internal resources.
  - **How it works:** Attackers use SSRF to generate excessive requests to internal services, overwhelming them or consuming excessive bandwidth and resources.
  - **Impact:** This can cause **service downtime**, making internal systems unavailable or slow.
- **Example Attack:**
  The attacker sends a flood of SSRF requests to an internal service, consuming CPU, memory, or network bandwidth until the service becomes unresponsive or crashes.

---

8. Accessing and Manipulating Admin Interfaces

- **Scenario:** Attackers use SSRF to gain access to internal **admin interfaces** that are not publicly available.
  - **How it works:** The attacker targets internal services running administrative tools or APIs. These interfaces are often protected by network-level restrictions but can be accessed via SSRF.
  - **Impact:** Gaining access to an **admin panel** can allow attackers to change configurations, escalate privileges, or retrieve sensitive information about the application.
- **Example Attack:**
  The attacker exploits SSRF to access an internal admin interface (`http://localhost:8080/admin`) and gains unauthorized control over the system.

---

9. Exploiting Vulnerable Internal Web Services (Webhooks or APIs)

- **Scenario:** SSRF can be used to trigger internal webhooks or APIs that may be vulnerable or improperly configured.
  - **How it works:** An attacker sends an SSRF request to an internal service or API that processes user input or provides sensitive operations (e.g., sending emails, accessing external resources).

    - **Impact:** If these services are not properly validated or authenticated, the attacker can **exploit** them for **unauthorized actions**, such as sending emails to external attackers, triggering internal system actions, or **accessing external resources**.
- **Example Attack:**
  The attacker targets a webhook endpoint (`http://localhost:5000/notify`) and sends a crafted SSRF request to trigger a sensitive internal operation, like sending an email with stolen credentials or accessing an external resource.

Summary of Common SSRF Attack Scenarios

- **Internal Services Access (Port Scanning):** Exploiting SSRF to interact with internal services like databases and admin panels.
- **Cloud Metadata Exposure:** Leveraging SSRF to retrieve cloud-specific metadata or credentials, gaining unauthorized cloud access.
- **Bypassing Firewalls and IP Restrictions:** Using SSRF to bypass IP whitelisting or firewall rules, accessing otherwise restricted internal systems.
- **Exploiting Internal Vulnerabilities:** Triggering remote code execution or exploiting internal vulnerabilities through SSRF.
- **Accessing Sensitive Files:** Accessing sensitive files like `/etc/passwd`, which could leak system information or credentials.
- **External Redirects to Malicious Servers:** Redirecting SSRF requests to external malicious servers controlled by the attacker.
- **DoS via Resource Exhaustion:** Overloading internal services with excessive requests, causing service outages.
- **Admin Interface Access:** Gaining unauthorized access to internal admin interfaces or control panels.
- **Vulnerable Web Services Exploitation:** Exploiting vulnerable internal webhooks or APIs using SSRF.

Each of these attack scenarios illustrates the versatility of SSRF in exploiting **internal network resources** or **misconfigurations** in web applications. Preventing SSRF requires a combination of input validation, restricting outgoing requests, and applying **principle of least privilege** to internal services. Would you like to explore mitigation techniques or see specific examples in more detail?